データベース第5回

第5章　SQL

SQL

- SQL(エスキューエル，スィークエル)
  - 国際標準リレーショナルデータベース言語
  - SQL-87, SQL-92, SQL:1999, SQL:2003, SQL:2006, SQL:2008, SQL:2011, SQL:2016, SQL:2019
  - 機能
    - データ定義とデータ操作
    - たくさん．．．
  - <問合せ指定>の基本構文図5.1参照（BNF記法）


```
<問合せ指定> ::=
    SELECT [ALL | DISTINCT] <選択リスト> <表式>
<選択リスト> ::=
    <値式> [{, <値式>}] | *
<表式> ::=
    <FROM 句>
    [<WHERE 句>]
    [<GROUP BY 句>]
    [<HAVING 句>]
<FROM 句> ::=
    FROM <表参照> [(, <表参照>)···]
<WHERE 句> ::=
    WHERE <探索条件>
<探索条件> ::=
    <ブール一次子> | <探索条件> OR <ブール一次子>
<ブール一次子> ::=
    <ブール因子> | <ブール一次子> AND <ブール因子>
<ブール因子> ::=
    [NOT] <ブール素項>
<ブール素項> ::=
    <述語> | (<探索条件>)
<GROUP BY 句> ::=
    GROUP BY <列指定> [(, <列指定>)···]
<HAVING 句> ::=
    HAVING <探索条件>
```


SQLの問合せ指定の基本形

SELECT　<値式$_1$>, <値式$_2$>, ･･･, <値式$_n$>
FROM　<表参照$_1$>, <表参照$_2$>, ･･･, <表参照$_m$>
WHERE　<検索条件>

SQL: Intro

- "S.Q.L." or "sequel"
- Supported by all major commercial database systems
- Standardized – many new features over time
- Interactive via GUI or prompt, or embedded in programs
- Declarative, based on relational algebra

SQL: Intro

Data Definition Language (DDL)

Create table ...

drop table ...

Data Manipulation Language (DML)

select

insert

delete

update

Other Commands

indexes, constraints, views, triggers, transactions, authorization, ...

SQL: Intro

The Basic SELECT Statement

```
Select  A1,A2,…,An      ← what to return
From    R1,R2, …,Rm     ← relations
Where   condition       ← combine filter
```

(3) (1) (2)

$$\Pi_{A_1,..,A_n}\left(\sigma_{condition}(R_1 \times .. \times R_m)\right)$$

社員-部門データベース

社員

社員番号	社員名	給与	所属
0650	鈴木一郎	50	K55
1508	浜崎アユ	40	K41
0231	宇田ひかる	60	K41
2034	別所幸治	40	K55
0713	小松百合子	60	K55

部門

部門番号	部門名	部門長
K55	データベース	0650
K41	ネットワーク	1508

1.単純質問
2.結合質問
3.入れ子型質問

単純質問

全社員の全データを見たい

SELECT　＊
FROM　社員

単純質問

社員の給与一覧を見たい（射影演算の基本）

(a)
SELECT 給与
FROM 社員

導出表：

給与
50
40
60
40
60

(b)
SELECT DISTINCT 給与
FROM 社員

導出表：

給与
50
40
60

単純質問

K55に所属している社員の社員番号，社員名，給与を見たい（選択演算と射影演算の基本）

SELECT　社員番号, 社員名, 給与
FROM　社員
WHERE　所属='K55'

単純質問

- WHERE句の述語には，NOT, AND, ORで結合した条件が記述可能

単純質問

K55に所属していて給与が50以上の社員の社員番号と社員名を知りたい

SELECT 社員番号, 社員名
FROM 社員
WHERE 所属='K55' AND 給与>=50

単純質問

社員の給与とその0.8掛けの値を知りたい

SELECT 社員番号, 社員名, 給与, 給与×0.8
FROM 社員

社員番号	社員名	給与	
0650	鈴木一郎	50	40
1508	浜崎あゆみ	40	32
0231	宇田ひかる	60	48
2034	別所幸治	40	32
0713	小松百合子	60	48

単純質問

- 比較演算子θに加えて，BETWEEN, IN, LIKE, NULL, EXISTSなどの述語が使用可能

単純質問

給与が40以上かつ50以下の社員の全データを求める

SELECT ＊
FROM 社員
WHERE 給与 BETWEEN 40 AND 50

単純質問

- ORDER BY, GROUP BY, HAVING
- 集約関数
  – COUNT, SUM, AVG, MAX, MIN

17

単純質問

SELECT 所属, AVG(給与)
FROM 社員
GROUP BY 所属
HAVING COUNT(*)>=3

社員

社員番号	社員名	給与	所属
0650	鈴木一郎	50	K55
1508	浜崎アユ	40	K41
0231	宇田ひかる	60	K41
2034	別所幸治	40	K55
0713	小松百合子	60	K55

所属	
K55	50

18

結合質問

データベース部に所属している社員の社員番号と社員名を求めよ

SELECT X.社員番号, X.社員名
FROM 社員 X, 部門 Y
WHERE X.所属=Y.部門番号
　　AND Y.部門名='データベース'

19

結合質問

上司よりも高給をとっている社員の社員番号と社員名を求めよ

SELECT X.社員番号, X.社員名
FROM 社員 X, 部門 Y, 社員 Z
WHERE X.所属=Y.部門番号
　AND Y.部門長=Z.社員番号
　AND X.給与>Z.給与

20

入れ子型質問

平均給与より高給を取っている社員データを求めよ

```
SELECT ＊
FROM 社員
WHERE 給与>
   (SELECT AVG(給与)
    FROM 社員)
```


入れ子型質問

直属の上司よりも高給を取っている社員の社員番号と社員名を求めよ

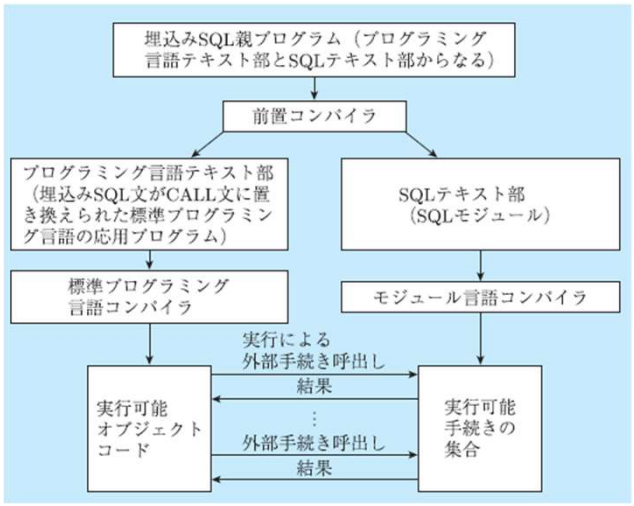
```
SELECT X.社員番号, X.社員名
FROM 社員 X
WHERE X.給与>
   (SELECT Z.給与
    FROM 部門 Y, 社員 Z
    WHERE X.所属=Y.部門番号
    AND Y.部門長=Z.社員番号)
```


SQLと計算完備性

- SQLはリレーショナル完備
  - リレーショナル代数と同等の表現能力
- SQLは計算完備ではない
  - プログラミング言語の表現能力より低い
- 埋込みSQLとSQL/PSM (SQL/Persistent Stored Module)

埋込みSQL

- プログラミング言語（特にコンパイラ型言語）からSQLを使う場合に使用
  - 計算完備なプログラミング言語から，関係完備なSQLを使う
  - （親）プログラミング言語で表のデータはそのままは扱えない．表の中の一つ一つのタップルを扱うために，カーソルを用いる
  - コンパイルした実行ファイルとSQLの処理部分は別になる．SQL処理部は，実行ファイル中から外部呼出しにより実行される（図5.4）



25

SQLと非ビジネスデータ処理

- SQLのオブジェクト指向拡張

- SQLのXML拡張

26

SQLによる問合せの記述

27

SQLによる問合せの記述

- SQLの基本的な書き方
- 条件（WHERE）の書き方
- 出力（SELECT）の書き方
- 順序付け（ORDER BY）
- グループ表（GROUP BY）
- 結合（JOIN）
- 集合演算
- 副問合せ（入れ子型質問）

28

SQLの記述方法

- 基本的な記述方法

```
SELECT   表.属性(値式), …
FROM     表, …
WHERE    条件式 AND …
```

  - SELECT 節
    - 問合せの結果として取り出す属性(値式)
  - FROM 節
    - どの表(テーブル)から検索するかを指定
  - WHERE 節
    - 検索の条件を指定

表の例

- 以降の記述例で使う表(リレーション)の例

科目

科目番号	科目名	単位数
001	データベース	2
002	システムプログラム	3
⋮	⋮	⋮

学生

学籍番号	氏名	専攻	住所
00001	山田一郎	情報工学	東京都×××
00002	鈴木明	情報工学	茨城県△△△
00003	佐藤花子	知識工学	京都府○○○
⋮	⋮	⋮	⋮

履修

科目番号	学籍番号	成績
001	00001	90
001	00002	80
002	00001	90
002	00003	70
⋮	⋮	⋮

実習課題

科目番号	課題番号	課題名
001	01	データモデリング
001	02	データベース設計
001	03	ＳＱＬ
002	01	Ｃプログラミング
002	02	システムコール
⋮	⋮	⋮

SQLによる問合せの記述

- SQLの基本的な書き方
- 条件(WHERE)の書き方
- 出力(SELECT)の書き方
- 順序付け(ORDER BY)
- グループ表(GROUP BY)
- 結合(JOIN)
- 集合演算
- 副問合せ(入れ子型質問)

条件(WHERE)の書き方

- 属性に関する条件を記述できる
  - =, <=, >=
  - 複数の条件を AND や OR で組み合わせることもできる
  - NOT で条件を反転することもできる

単純な質問の例（1）

Q.科目番号001の履修者の学籍番号と成績の一覧

```
SELECT   履修.学籍番号, 履修.成績
FROM     履修
WHERE    履修.科目番号 = '001'
```

- 表名が一意に決まる時は表名は省略できる

```
SELECT   学籍番号, 成績
FROM     履修
WHERE    科目番号 = '001'
```


単純な質問の例（1）

```
SELECT   学籍番号, 成績
FROM     履修
WHERE    科目番号 = '001'
```

履修

科目番号	学籍番号	成績
001	00001	90
001	00002	80
002	00001	90
002		70

出力される表

学籍番号	成績
00001	90
00002	80

① FROM節に書かれた表のうち，WHERE節の条件をみたすデータ（行）が出力対象となる

② SELECT節に書かれた属性（列）が出力対象となる

単純な質問の例（2）

- ANDを使う例

Q.学籍番号00100の学生の科目番号5の成績

```
SELECT   成績
FROM     履修
WHERE    科目番号 = '005' AND 学籍番号='00100'
```

- WHERE節を省略した例

Q. 全科目の科目名と単位数の一覧

```
SELECT   科目名, 単位数
FROM     科目
```


述語

- 条件を記述するために使える述語
  - BETWEEN x AND y ・・・ 属性値の範囲を指定
  - IN t ・・・ 表 t に含まれるかどうかを判定
  - LIKE x ・・・ 文字列の部分一致，x を含むか
  - NULL ・・・ 空値かどうかを判定
  - EXISTS t ・・・表 t が空集合でない場合に真

述語の例

- Q. 科目番号001の科目で80点から90点の成績の学籍番号の一覧

```
SELECT   学籍番号
FROM     履修
WHERE    科目番号 = '001' AND
         成績 BETWEEN 80 AND 90
```

```
SELECT   学籍番号
FROM     履修
WHERE    科目番号 = '001' AND
         成績 >= 80 AND 成績 <= 90
```

※ どちらでも同じ（どちらの書き方でも良い）

述語の例

- Q. 名前に「子」の文字を含む学生の学籍番号と氏名の一覧

```
SELECT   学籍番号, 氏名
FROM     学生
WHERE    氏名 LIKE '子'
```


SQLによる問合せの記述

- SQLの基本的な書き方
- 条件（WHERE）の書き方
- 出力（SELECT）の書き方
- 順序付け（ORDER BY）
- グループ表（GROUP BY）
- 結合（JOIN）
- 集合演算
- 副問合せ（入れ子型質問）

全属性を出力

- 指定した表の全部の列を問合せ結果に含めたい場合は，＊で省略できる

Q.単位数が3単位以上の科目の科目番号，科目名，単位数の一覧

– 科目（科目番号，科目名，単位数）

```
SELECT   *
FROM     科目
WHERE    単位数 >= 3
```

重複の除去

- DISTINCT指定
  - 重複した行を除去するための指定
  - SQLでは，重複は自動的に除去されない

Q. 全科目の科目名と単位数の一覧

```
SELECT   DISTINCT 科目名, 単位数
FROM     科目
```

41

集約関数

- 検索結果の表に対して集計演算を行う
  - COUNT（行数），SUM（合計値），AVG（平均値），MAX（最大値），MIN（最小値）

Q. 科目番号 001 の平均点，最小点，最高点を表示

```
SELECT   AVG(成績), MIN(成績), MAX(成績)
FROM     履修
WHERE    科目番号 = '001'
```

42

集約関数の例

Q. 科目番号 001 の受講者の人数を表示

```
SELECT   COUNT(*)
FROM     履修
WHERE    科目番号 = '001'
```

- COUNTは，出力されるデータの属性ではなく，データの数の合計を計算する，特殊な集約関数
  - 属性を指定する意味はないので，関数の引数には，全属性を表す＊を記述

43

集約関数

- COUNTと他の集約関数の違いに注意

履修

科目番号	学籍番号	成績
001	00001	90
001	00002	80
002	00001	90
002	00002	70

属性値の合計なので「170」

```
SELECT   SUM(成績)
FROM     履修
WHERE    科目番号 = '001'
```

データ数の合計なので「2」

```
SELECT   COUNT(*)
FROM     履修
WHERE    科目番号 = '001'
```

44

SQLによる問合せの記述

- SQLの基本的な書き方
- 条件（WHERE）の書き方
- 出力（SELECT）の書き方
- 順序付け（ORDER BY）
- グループ表（GROUP BY）
- 結合（JOIN）
- 集合演算
- 副問合せ（入れ子型質問）

順序付け（ORDER BY）

- ORDER BYの例

Q.科目番号 005 を履修した学生の学籍番号と成績の一覧を，成績の良い順番に出力

```
SELECT      学籍番号, 成績
FROM        履修
WHERE       科目番号 = '005'
ORDER BY    成績 DESC
```

– ASC（ascendant）・・・昇順，小から大へ

– DESC（descendant）・・・降順，大から小へ

SQLによる問合せの記述

- SQLの基本的な書き方
- 条件（WHERE）の書き方
- 出力（SELECT）の書き方
- 順序付け（ORDER BY）
- グループ表（GROUP BY）
- 結合（JOIN）
- 集合演算
- 副問合せ（入れ子型質問）

グループ表（GROUP BY）

- GROUP BY
  - 同一属性値の行をグループ化できる
  - 集約関数は，全データではなく，各グループごとの全データに適用されるようになる

Q. 全科目について科目番号と平均点の一覧

```
SELECT      科目番号, AVG（成績）
FROM        履修
GROUP BY    科目番号
```

グループ表

- グループ表の例

集約演算は，各グループごとに適用されることに注意

科目番号	学籍番号	成績
001	001001	65
001	001002	75
001	001004	70
002	001002	80
002	001003	60
002	001004	90
002	001005	70
003	001004	70
…	…	…

各グループをひとつのデータ（行）として出力

科目番号	
001	70
002	75
…	…

グループ表 ＋ グループ選択

- GROUP BY + HAVING
  - HAVING によりグループの選択の条件を指定

Q. 履修者が3名以上の科目の科目番号，履修者数，平均点の一覧

```
SELECT    科目番号, COUNT(*), AVG(成績)
FROM      履修
GROUP BY  科目番号
HAVING    COUNT(*) >= 3
```


グループ表

- グループ表の例

科目番号	学籍番号	成績
001	001001	65
001	001002	75
001	001004	70
002	001002	80
002	001003	60
002	001004	90
002	001005	70
003	001004	70
…	…	…

WHERE

条件を満たすグループのデータのみ出力

科目番号		
001	3	70
002	4	75
…		…

GROUP BY　　HAVING

GROUP BY の適用順序

```
FROM      ……（入力とするテーブル）
WHERE     ……（出力するデータを選択）
GROUP BY  ……（出力されたデータをグループ化）
HAVING    ……（出力するグループを選択）
SELECT    ……（出力する属性）
```

- FROM 節のテーブルの各データの組合せから，WHERE 節で書かれている条件を満たす組を選択
- 選択されたデータに対して，GROUP BY 節に書かれている属性の値が同じもの同士でグループ化
- 各グループごとに，HAVING 節に書かれている条件を満たすかどうか判定し，条件を満たすもののみを選択
- 選択されたデータ or グループの属性のうち，SELECT 節に書かれているものを出力

GROUP BY 使用時の注意

- GROUP BY 使用時の注意

```
SELECT    部門名, AVG(年齢)
FROM      従業員, 部門
WHERE     従業員.部門番号 = 部門.部門番号
GROUP BY  部門番号
HAVING    COUNT(*) > 2
```

  - GROUP BY 節があるときは，各グループが出力の単位となるので，SELECT 節や HAVING 節にはグループで有効な属性（GROUP BY に使った属性）や集約関数しか書けない

SQLによる問合せの記述

- SQLの基本的な書き方
- 条件（WHERE）の書き方
- 出力（SELECT）の書き方
- 順序付け（ORDER BY）
- グループ表（GROUP BY）
- 結合（JOIN）
- 集合演算
- 副問合せ（入れ子型質問）

結合質問

- 結合質問（Join Query）
- 複数の表を組み合わせた質問（表同士の結合が起こる質問）

結合

Q.学籍番号 00001 の学生が履修した科目の科目番号，科目名，成績の一覧

```
SELECT   科目.科目番号, 科目名, 成績
FROM     科目, 履修
WHERE    科目.科目番号 = 履修.科目番号  AND
         学籍番号 = ' 00001'
```

  - 科目と履修を科目番号で等結合
    科目（科目番号, 科目名, 単位数）
    履修（科目番号, 学籍番号, 成績）
    →（科目.科目番号, 科目名, 単位数,
    履修.科目番号, 学籍番号, 成績）

結合の例

```
SELECT   科目.科目番号, 科目名, 成績
FROM     科目, 履修
WHERE    科目.科目番号 = 履修.科目番号 AND
         学籍番号 = '00001'
```

履修

科目番号	学籍番号	成績
001	00001	90
001	00002	80
002	00001	90
002	00003	70

×

科目

科目番号	科目名	単位数
001	データベース	2
002	グラフィックス	2
003	プログラミング	2

2つの表の各データ同士の組合せのうち，条件を満たすデータ同士の組合せが出力される
（この例では，4×3＝12通りのデータのうち，2つが出力される）

結合の例（続き）

```
SELECT   科目.科目番号, 科目名, 成績
FROM     科目, 履修
WHERE    科目.科目番号 = 履修.科目番号 AND
         学籍番号 = '00001'
```

履修.科目番号	学籍番号	成績	科目.科目番号	科目名	単位数
001	00001	90	001	データベース	2
002	00001	80	002	グラフィックス	2

⇩ SELECT節で指定された属性のみを出力

科目.科目番号	科目名	成績
001	データベース	90
002	グラフィックス	80

自然結合

- NATURAL JOIN
  - 同じ質問を NATURAL JOIN を使っても書ける

```
SELECT   科目番号, 科目名, 成績
FROM     科目 NATURAL JOIN 履修
WHERE    学籍番号 = '00001'
```

  - 科目と履修を自然結合
    科目(<u>科目番号</u>, 科目名, 単位数)
    履修(<u>科目番号</u>, 学籍番号, 成績)
    →（科目番号, 科目名, 単位数, 学籍番号, 成績）

自己結合

- 同一の表同士の結合
  - 同一の表を複数使うときに，それぞれの表を区別するために，表に名前をつけることができる
  - 例:
    - FROM 学生 AS x, 学生 AS y

自己結合の例

Q.科目番号 001 の科目に関して，
学籍番号 001001 の学生よりも成績の良かった学生の学籍番号の一覧

```
SELECT   y.学籍番号
FROM     履修 AS x, 履修 AS y
WHERE    x.科目番号 = '001' AND x.学籍番号 = '001001'
         AND y.科目番号 = '001' AND y.成績 > x.成績
```

– x は科目番号001，学籍番号001001の履修を指定
– y は x よりも成績の良い履修を指定

61

自己結合の例（続き）

```
WHERE    x.科目番号 = '001' AND x.学籍番号 = '001001'
         AND y.科目番号 = '001' AND y.成績 > x.成績
```

履修 AS x

科目番号	学籍番号	成績
001	001001	60
001	001002	75
001	001004	70
001	001005	50
002	001003	60
002	001004	90
003	001004	70
…	…	…

履修 AS y

科目番号	学籍番号	成績
001	001001	60
001	001002	75
001	001004	70
001	001005	50
002	001003	60
002	001004	90
003	001004	70
…	…	…

62

SQLによる問合せの記述

- SQLの基本的な書き方
- 条件（WHERE）の書き方
- 出力（SELECT）の書き方
- 順序付け（ORDER BY）
- グループ表（GROUP BY）
- 結合（JOIN）
- 集合演算
- 副問合せ（入れ子型質問）

63

集合演算

- 複数の表同士の集合演算
  - UNION（和）
  - EXCEPT（差）
  - INTERSECT（共通部分）
- 表の各列のデータは一致する必要がある
- 集合演算では，自動的に重複する行は除去される
  - 重複を除去しない集合演算も存在
  - UNION ALL, EXCEPT ALL, INTERSECT ALL

64

集合演算の例

- UNION（和集合）の例

Q.実習課題があるか，単位数が5単位以上の科目の科目番号，科目名，単位数の一覧

```
SELECT   科目.*
FROM     科目, 実習課題
WHERE    科目.科目番号 = 実習課題.科目番号
UNION
SELECT   *
FROM     科目
WHERE    単位数 >= 5
```


集合演算の例

- EXCEPT（差集合）の例

Q.実習課題のない科目の科目番号，科目名の一覧

```
SELECT   科目番号, 科目名
FROM     科目
EXCEPT
SELECT   科目.科目番号, 科目名
FROM     科目, 実習課題
WHERE    科目.科目番号 = 実習課題.科目番号
```


SQLによる問合せの記述

- SQLの基本的な書き方
- 条件（WHERE）の書き方
- 出力（SELECT）の書き方
- 順序付け（ORDER BY）
- グループ表（GROUP BY）
- 結合（JOIN）
- 集合演算
- 副問合せ（入れ子型質問）

副問合せ（入れ子型質問）

- 入れ子型質問（nested query）
  - SQLでは副問合せ（subquery）という用語で定義されている
- 問合せの結果に対してさらに問合せを行うことができる

入れ子型質問

- 入れ子型質問の例

Q.科目番号 001 の成績が平均点よりも高い学生の学籍番号と成績の一覧

```
SELECT   学籍番号, 成績
FROM     履修
WHERE    科目番号 = '001' AND
         成績 >
   (SELECT   AVG(成績)
    FROM     履修
    WHERE    科目番号 = '001' )
```


入れ子型質問の例

科目番号	学籍番号	成績
001	001001	60
001	001002	76
001	001004	70
001	001005	50
002	001003	60
002	001004	90
003	001004	70
…	…	…

```
SELECT   学籍番号, 成績
FROM     履修
WHERE    科目番号 = '001' AND
         成績 >
   (SELECT   AVG(成績)
    FROM     履修
    WHERE    科目番号 = '001' )
```

⇨ 64

内側の問合せを最初に一度だけ実行

入れ子型質問の例（続き）

科目番号	学籍番号	成績
001	001001	60
001	001002	76
001	001004	70
001	001005	50
002	001003	60
002	001004	90
003	001004	70
…	…	…

```
SELECT   学籍番号, 成績
FROM     履修
WHERE    科目番号 = '001' AND
         成績 >
   (SELECT   AVG(成績)
    FROM     履修
    WHERE    科目番号 = '001' )
```

> 64

外側の問合せを実行